ADOBE AIR FOR ANDROID –SOVELLUSKEHITYS

Jani Palovuori

Opinnäytetyö
Joulukuu 2011
Tietojenkäsittelyn koulutusohjelma
Digitaalisen median ja
ohjelmistotekniikan
suuntautumisvaihtoehto
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Digitaalisen median ja ohjelmistotekniikan suuntautumisvaihtoehto

PALOVUORI, JANI: Adobe AIR for Android –sovelluskehitys

Opinnäytetyö 35 sivua
Joulukuu 2011

Tässä opinnäytetyössä käsitellään sovelluskehitystä Android-laitteille Adoben AIR for Android -tekniikkaa hyväksi käyttäen. Tavoitteena on esitellä sovelluskehityksen periaatteita Adoben Flash-tekniikkaan pohjautuvan AIR for Android -teknologian kannalta. Työ on tehty Yleisradio Oy:n toimeksiantona ja toimii johdatuksena AIR-sovelluksen kehittämiseen AIR for Android -teknologialla.

Työssä esitellään yleisesti rikkaat internet-sovellukset ja käydään tarkemmin läpi Adoben AIR-teknologia sekä sen toimintaperiaatteet. Työssä esitellään kehityksessä käytettäviä työkaluja ja käydään läpi niiden käyttöönotto sekä asennus. Lopuksi pohditaan hieman AIR for Android -sovelluskehityksen mahdollisuuksia ja tulevaisuutta.

Opinnäytetyön tarkoituksena on auttaa AIR for Android -teknologiaan tutustuvia opiskelijoita. Se pyrkii antamaan kehityksessä tarvittavia taustatietoja ja käymään läpi AIR for Android -sovelluskehityksen erityispiirteitä.

Asiasanat: Android, Adobe AIR, sovelluskehitys

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of digital media and software engineering

PALOVUORI, JANI: Software Development Using Adobe AIR for Android

Bachelor’s thesis 35 pages
December 2011

This thesis discusses mobile Android development using the AIR for Android technology. The objective is to examine and introduce the AIR for Android development, which is based on Adobe's Flash technology. The thesis is an assignment from the Finnish Broadcasting Company Yleisradio and it serves as an introduction to mobile development using the AIR for Android technology.

The thesis explains Rich Internet Applications in general and focuses more closely on the Adobe AIR technology. The thesis introduces and examines the installation of the tools used in developing Adobe AIR for Android applications. Finally, some thoughts will be presented about the possibilities and future of the AIR for Android applications.

The aim of this thesis is to help students who wish to familiarize themselves with the AIR for Android technology. It aims to provide background knowledge needed in the development process and also to explain some of the characteristics of the AIR for Android software development.

Keywords: Android, Adobe AIR, software development

SISÄLTÖ

1 JOHDANTO

Erilaisten älypuhelinten yleistyessä huimaa vauhtia, kulkee Internet-yhteys yhä useammin jokaisen mukana kaikkialle. Monenlaiset kehittyneemmät web -sovellukset ovat myös jo pitkään muovanneet ihmisten käsityksiä siitä, millaisia palveluita on mahdollisuus tarjota jo pelkästään tutun Internet-selaimen kautta. Tämänkaltaisia kehittyneempiä web-sovelluksia kutsutaan nykyisin rikkaiksi internet-sovelluksiksi.

Mobiililaitteiden yleistymisen vuoksi on myös yhä tärkeämpää pyrkiä tavoittamaan ihmiset kotitietokoneiden lisäksi erilaisten mukana kulkevien mobiililaitteiden avulla. Palveluiden monistaminen useaan eri versioon voi kuitenkin pahimmassa tapauksessa vaatia merkittäviä määriä lisää resursseja, ja lisätä näin kustannuksia.

Työskentelin vuoden vaihteessa 2010-2011 Yleisradio Oy:n alaisuudessa, uuden lastenohjelman parissa. Yksi tehtävistäni oli tutkia ja toteuttaa mobiiliversio jo olemassa olevan, Flash-tekniikalla toteutetun verkkosovelluksen pohjalta. Tutustuin työssäni Adoben AIR for Android -teknologiaan, joka mahdollistaa Flash-tekniikalla toteutetun sovelluksen ajamisen natiivin sovelluksen tavoin Android-käyttöjärjestelmän omaavassa mobiililaitteessa. Työhöni kuului paljon tutkivaa kehitystä ja uuden tekniikan opiskelua, jota pyrin nyt hyödyntämään opinnäytetyössäni.

Opinnäytetyöni toimii johdatuksena mobiilikehittämiseen Adoben AIR for Android -tekniikkaa hyödyntäen. Esittelen aluksi rikkaiden internet-sovellusten taustaa, kehitystä, eri muotoja ja ominaispiirteitä. Tämän jälkeen esittelen tarkemmin Adobe AIR -sovellustekniikkaa ja käyn läpi käyttöönoton ja asennuksen eri vaiheet. Työni lopussa esitän yhteenvedon, ja pohdin AIR for Android -mobiilikehitystä yleisellä tasolla.

2 RIKKAAT INTERNET-SOVELLUKSET (RIA)

Rikkaat internet-sovellukset (engl. RIA, Rich Internet Applications) ovat internet -sovelluksia, jotka pyrkivät toiminnallisuudeltaan vastamaan tavallisia työpöytäsovelluksia. Teknisesti niiden on tarkoitus tarjota käyttäjälle tavallisia web-sovelluksia monipuolisempi ja viehättävämpi käyttökokemus.

Shockwave- ja Flash-tekniikoiden alkuperäinen kehittäjä, Macromedia, esitteli termin ensimmäisen kerran vuonna 2002 (Allaire 2002, 1). Tuolloin rikkaille internet-sovelluksille määriteltiin tietynlaiset vaatimukset, joita noudattamalla niiden oli tarkoitus puuttua siihen aikaan suosiotaan edelleen jyrkästi kasvattavien internet-sovellusten ongelmiin.

2.1 Rakenne

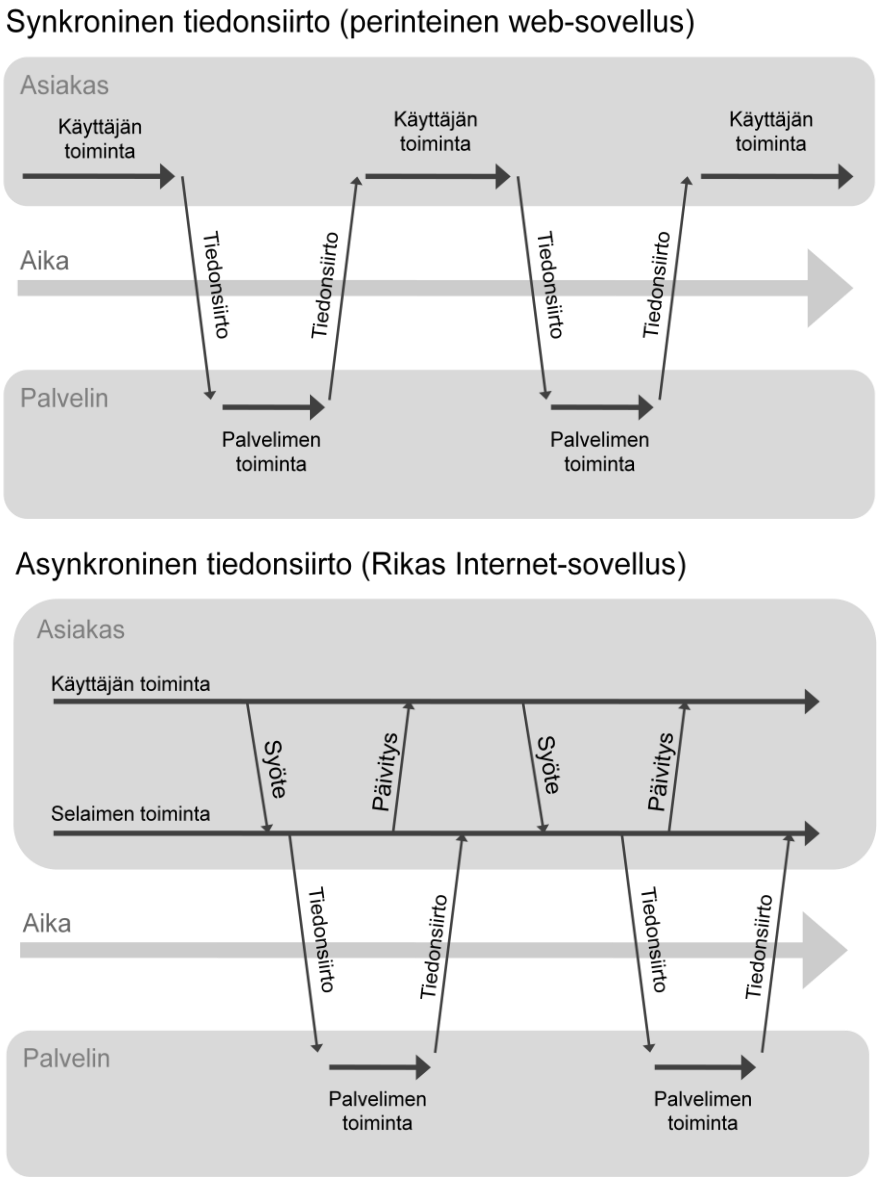
2.1.1 Käyttöliittymä ja käytettävyys

Rikkaan internet-sovelluksen käyttöliittymä pyrkii vastaamaan työpöytäsovelluksen käyttöliittymää tarjoamalla toiminnaltaan monipuolisia käyttöliittymäkomponentteja. Rikkaat internet-sovellukset mahdollistavat myös tavallisiin web-sovelluksiin verrattuna vaativimpien toimintojen suorittamisen.

Merkittävää rikkaissa internet-sovelluksissa on myös niiden tapa toteuttaa tiedonsiirto asiakkaan ja palvelimen välillä. Sovellukset näet vähentävät siirrettävän tiedon määrää siirtämällä sekä esityslogiikan että mahdollisuuksien mukaan myös liiketoimintalogiikan asiakaspuolen prosessoitavaksi.

2.1.2 Asynkroninen tiedonsiirto

Rikkaan Internet-sovelluksen kommunikointi palvelimen kanssa tapahtuu vain silloin, kun sovellus joko lähettää tietoa tai tarvitsee sitä. Asynkronisten tiedonsiirtopyyntöjen avulla sovellus voi päivittää vain tarvittavan osan itsestään, sen sijaan että kaikki tieto ladattaisiin palvelimelta uudestaan (kuvio 1). Esimerkkinä tästä on henkilöhakemisto, johon käyttäjä suorittaa haun: asynkronisella pyynnöllä sovellus hankkii palvelimelta ainoastaan haun tuloksen, ja päivittää tulokset käyttöliittymään.



KUVIO 1. Synkroninen tiedonsiirtotapa verrattuna asynkroniseen tiedonsiirtoon (Garret 2005, muokattu).

Verrattuna synkroniseen tiedonsiirtoon, luo asynkroninen tiedonsiirto parantuneen käytettävyyden lisäksi myös tiettyjä haasteita. Vaikka asiakkaan ja palvelimen välillä siirrettävän tiedon määrä vähenee, tekee asiakassovellus asynkronisessa tiedonsiirrossa useampia pyyntöjä palvelimelle. Useat samaan aikaan prosessoitavat pyynnöt rasittavat palvelinta, ja voivat täten aiheuttaa hidastumista sovelluksessa. Lähetetyt pyynnöt eivät myöskään välttämättä palaudu takaisin asiakassovellukseen samassa järjestyksessä kuin ne lähetettiin.

Edellä mainitut asiat tulee ottaa huomioon sovellusta kehitettäessä. Yksi ratkaisu on toteuttaa pyynnöille jonkinlainen jonotuslogiikka, joka säätelee lähettävien pyyntöjen määrää suhteessa siihen, kuinka aikaisemmat pyynnöt ovat palautuneet palvelimelta takaisin asiakassovellukseen.

2.2 Sovellustyypit

Rikkaita Internet-sovelluksia voidaan toteuttaa teknisesti usealla eri tavalla. Seuraavaksi käyn lyhyesti läpi eri toteutustekniikat.

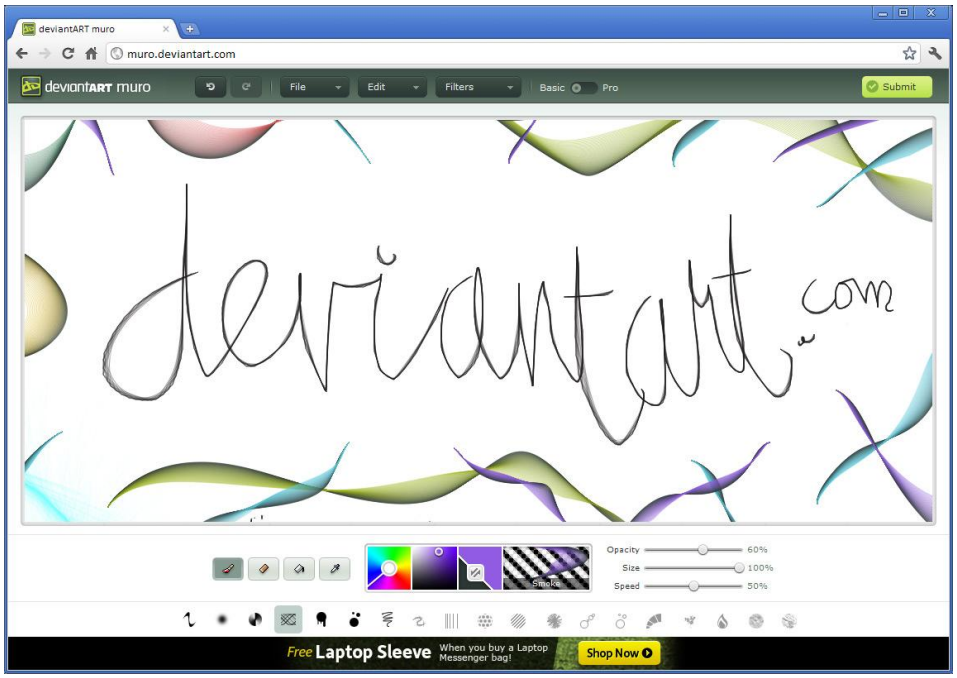
2.2.1 Internet-selaimessa toimiva sovellus

Suoraan selaimessa toimivan sovelluksen liiketoimintalogiikka toteutetaan selaimen tulkkaamalla komentosarjakielellä (yleisimmin JavaScript) ja käyttöliittymä HTML-merkintäkielellä sekä porrastetuilla tyyliarkeilla (engl. CSS, Cascading Style Sheets). Puhuttaessa tämänkaltaisista sovelluksista, käytetään nykyisin vakiintunutta ilmaisua Ajax-sovellukset (engl. Ajax, Asynchronous JavaScript and XML).

Ajax-sovellukset hyödyntävät XMLHttpRequest-oliota, joka mahdollistaa asynkronisen tiedonsiirron selaimen ja palvelimen välillä. Ajax-termi ei kuitenkaan kuvaa pelkästään yhtä tekniikkaa, vaan sen alle niputetaan JavaScript-, HTML- ja CSS- tekniikoiden lisäksi edellä mainittu XMLHttpRequest-olio sekä XML-merkintäkieli (engl. XML, Extensible Markup Language). XML-merkintäkieltä käytetään yleisimpänä muotona datan siirtämiseen takaisin palvelimelta asiakassovellukseen. (Garret 2005.)

Ajax-sovellukset toimivat suoraan internet-selaimessa, ilman erityisiä toimia käyttäjältä (kuvio 2). Tämän johdosta ne ovat saavutettavuudeltaan nyt läpikäytävistä sovelluksista korkeimmalla tasolla. Myös käytettävyys on Ajax -sovelluksissa hyvä, mutta ne eivät toiminnoiltaan aivan yllä muiden sovellustyyppien tasolle.

HTML-merkintäkielen uusin versio (HTML5) tosin määrittelee monta uutta ominaisuutta, jotka tuovat tämänkaltaiset sovellukset lähemmäksi seuraavaksi läpikäytäviä, selaimeen asennettavaan liitännäiseen perustuvia rikkaita internet -sovelluksia.



KUVIO 2. deviantArt Muro on selaimessa toimiva piirto-ohjelma. Sovellus hyödyntää HTML5-tekniikkaa, eikä vaadi erillistä asennusta.

2.2.2 Selaimeen asennettavalla liitännäisellä toimiva sovellus

Erilliseen, yleensä kolmannen osapuolen toteuttamaan selainliitännäiseen perustuvat sovellukset mahdollistavat monipuolisemman ja visuaalisesti näyttävämmän käyttökokemuksen (kuvio 3). Esimerkkeinä selaimeen asennettavista liitännäisistä mainittakoon kolme suosituinta: Adobe Flash Player, Sun Microsystems JavaFX ja Microsoft Silverlight (Rich Internet Application Market Share 2011).

Siinä missä liitännäiseen perustuvat sovellukset mahdollistavat paremman käyttökokemuksen, on niiden saavutettavuus riippuvainen siitä, onko käyttäjä asentanut Internet-selaimeensa tarvittavan liitännäisen. Tämä lieneekin liitännäisiin perustuvien sovellusten merkittävin ongelma.

KUVIO 3. Sumo Paint on selaimessa toimiva kuvankäsittelyohjelma. Sumo Paint on toteutettu Adobe Flash Player –tekniikkaa hyödyntäen.

2.2.3 Työpöytäsovelluksina ajettavat Rikkaat Internet-sovellukset

Kolmas sovellustyyppi edustaa ensimmäisenä läpikäytyyn, komentosarjakielellä toteutettuun, sovellukseen verrattuna toista ääripäätä. Tämänkaltaiset sovellukset asennetaan käyttäjän koneelle työpöytäsovellusten tavoin ja suoritetaan selaimen sijasta omassa erityisessä ajoympäristössään (kuvio 4). Esimerkkeinä mainittakoon Java Web Start -sovellukset ja tässä työssä tarkemmin läpikäytävät Adobe AIR -sovellukset.

Omassa ajoympäristössään ajettavat sovellukset mahdollistavat muun muassa sovelluksen pääsyn käyttäjän paikallisiin resursseihin, joskin tämä oikeus on yleensä rajattu eräänlaisen hiekkalaatikko-ympäristön (engl. Sandbox) avulla. Esittelen Adobe AIR -sovellusten hiekkalaatikko-ympäristöä tarkemmin työni luvussa 3.4 Tietoturva.

Työpöytäsovelluksina ajettavat rikkaat internet-sovellukset mahdollistavat kenties monipuolisimman ja käytettävyydeltään parhaan käyttökokemuksen, mutta vaativat käyttäjää asentamaan tietokoneelleen erillisen ajoympäristön ja kärsivät täten saavutettavuudessaan.

KUVIO 4. Tweetdeck-sovelluksen työpöytäversio hyödyntää Adobe AIR-teknologiaa. Tweetdeck on sovellus sosiaalisen median palveluiden hallintaan.

3 ADOBE AIR

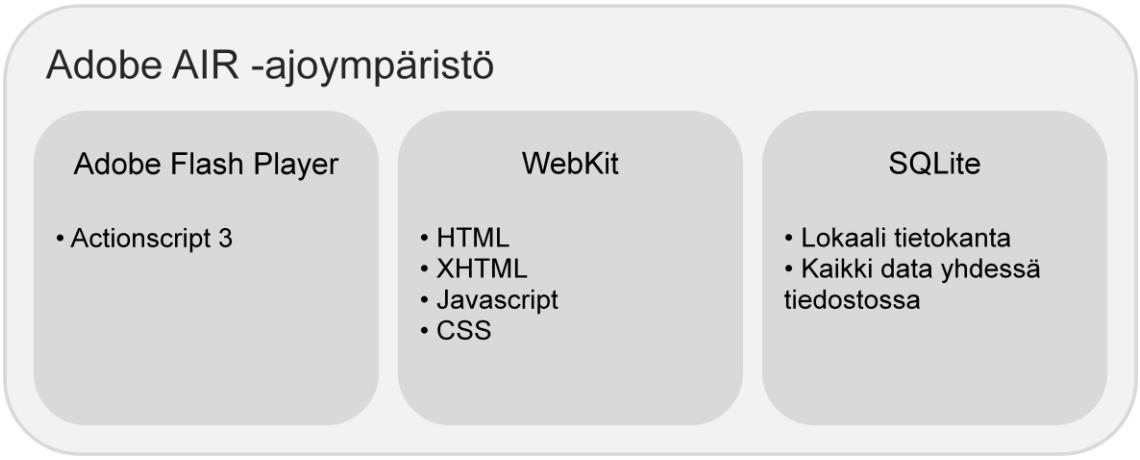
3.1 Yleistä

Adobe AIR on Adoben julkaisema erillinen ajoympäristö Adobe Flash-, Adobe Flex-, ja Ajax-tekniikoilla toteutetuille rikkaille internet-sovelluksille. Ensimmäinen versio julkaistiin vuonna 2008, ja tätä opinnäytetyötä kirjoitettaessa ajoympäristöstä julkaistiin versio 3.0. Adobe AIR tarjosi ilmestyessään mahdollisuuden käyttää jo olemassa olevaa Actionscript -lähdekoodia uusien, työpöydälle asennettavien sovelluksien toteuttamisessa.

Aluksi Adobe AIR –tekniikka tuki Microsoft Windows ja Mac Os X -käyttöjärjestelmiä, ja tuki Linux-käyttöjärjestelmille julkaistiin versiossa 1.5. Vuonna 2011 julkaistussa 2.7-versiossa Linux-tuki kuitenkin lopetettiin, koska Adoben mukaan Linux-version lataukset julkaisun alusta asti edustivat vain puolta prosenttia kokonaislatauksista (Adobe AIR / Frequently Asked Questions 2011). Uusien versioiden myötä tuki Adobe AIR -ajoympäristölle on laajentunut kattamaan tietokoneiden lisäksi myös lukuisia älypuhelinalustoja ja jopa televisioita (Flash Platform for TV 2011).

3.2 Ajoympäristö

Adobe AIR -ajoympäristö koostuu useasta eri komponentista. Kaikki näistä ovat, Adoben omaa Flash Player -tekniikkaa lukuun ottamatta, avoimen lähdekoodin teknologioita. Ajoympäristön keskeisimmät teknologiat ovat Adobe Flash Player, WebKit ja SQLite (kuvio 4). AIR -sovelluksia voidaan siis kehittää käyttäen joko web-kehittäjille tuttuja tekniikoita, tai Adobe Flash -sovelluksista tuttua ActionScript 3 –ohjelmointikieltä.



KUVIO 4. Adobe AIR -ajoympäristön eri komponentit.

3.2.1 Adobe Flash Player

Adobe AIR on kehitetty Adobe Flash Player -teknologian pohjalta. Tämä tarkoittaa sitä, että AIR-sovelluksia voidaan kehittää ActionScript 3 -ohjelmointikielellä. Myös kaikki Adoben Flash Player -teknologian ohjelmointirajapinnat ovat käytössä AIR-sovelluksia toteutettaessa, osa laajennettuna tai muutoin kehittyneempänä (Chambers, Dura, Georgita, Hoyt 2008, 8).

Flash Player -teknologia mahdollistaa myös monipuolisen grafiikan ja animaation esittämisen. Sen avulla sovelluksessa voidaan esittää vektorigrafiikkaa, videota, ääntä, kuvia ja 3D-animaatiota. Flash Playerin ohjelmointirajapintoja on mahdollista hyödyntää myös web-tekniikoita käytettäessä (Chambers ym. 2008, 8).

3.2.2 WebKit -selainmoottori

AIR-sovellukset käyttävät myös avoimen lähdekoodin WebKit -selainmoottoria. Selainmoottorin tehtävänä on HTML- ja CSS- tekniikoiden tulkitseminen ja niiden muodostaman kokonaisuuden esittäminen websivustona. WebKit -selainmoottori on käytössä esimerkiksi Google Chrome ja Apple Safari -selaimissa. Adobe AIR -ajoympäristössä WebKit-selainmoottori mahdollistaa muun muassa HTML-, JavaScript- ja CSS -tekniikoiden käyttämisen sovelluskehityksessä. AIR-sovelluksia onkin mahdollista kehittää myös pelkästään web-tekniikoita hyväksi käyttäen (Chambers ym. 2008, 7).

3.2.3 SQLite relaatiotietokantajärjestelmä

Adobe AIR ajoympäristö sisältää myös sulautetun SQLite relaatiotietokantajärjestelmän, jota sovellukset voivat hyödyntää. Koska SQLite on linkitetty suoraan AIR-sovellukseen, se ei tarvitse ajurin kautta toimivaa yhteyttä eikä erillistä tietokantapalvelinta. Kaikki tietokannat ja data tallennetaan yhteen tiedostoon tietokoneella (Distinctive Features of SQLite 2011).

Useimmista SQL-tietokantajärjestelmistä poiketen SQLite -tietokannan tietotyypin määrittely tehdään arvokohtaisesti, sen sijaan että tietotyyppi määriteltäisiin saraketta kohden (Distinctive Features of SQLite 2011). Tietotyyppi siis määritellään aina talletettavan arvon ominaisuutena, ja näin ollen sarakkeet voivat sisältää tyypiltään eroavia arvoja.

3.3 AIR for Android

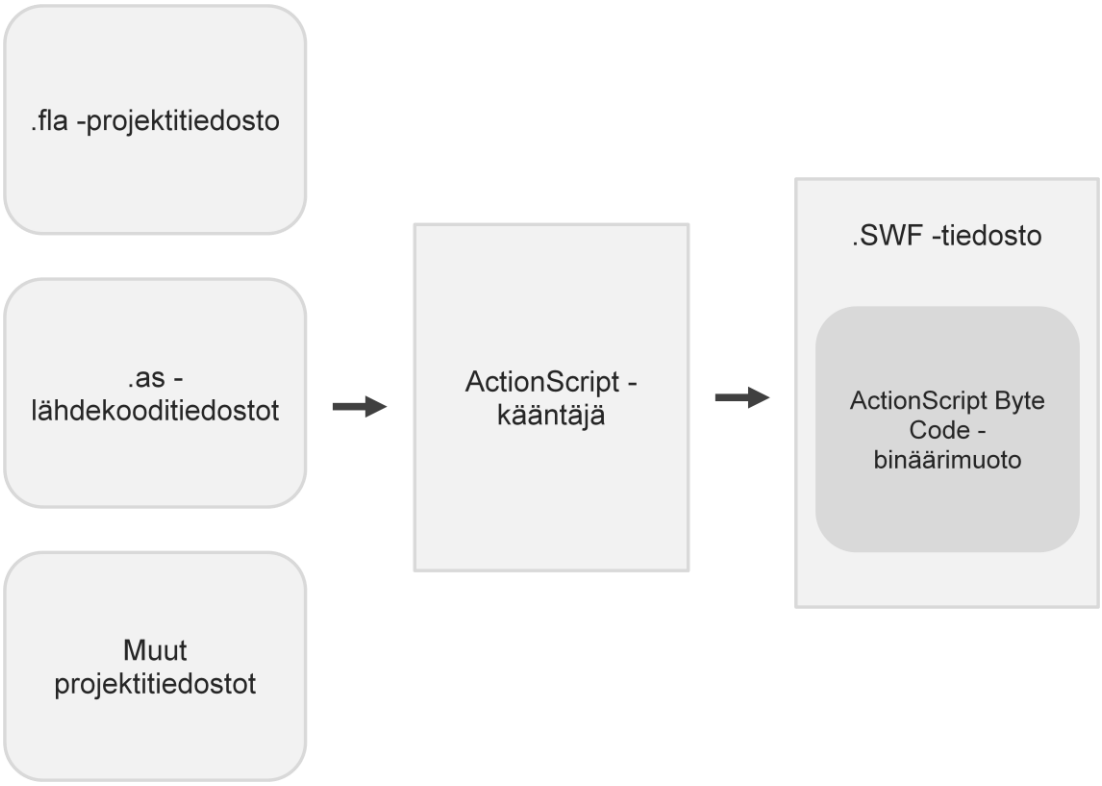
3.3.1 Yleistä

Adobe AIR for Android julkistettiin vuoden 2010 helmikuussa, ja tuki Android -laitteille julkaistiin saman vuoden kesäkuussa, AIR-ajoympäristön versionumerossa 2.0. Android–laitteille kehittäminen mahdollistettiin aluksi Flash Professional -sovellukseen asennettavan liitännäisen avulla, jonka pystyi hankkimaan erillisen prerelease-ohjelman kautta. Nykyisin kehittämisen mahdollistava komponentti löytyy integroituna uusimman CS 5.5 –version Flash Professionalista.

3.3.2 Android-laitteelle kehittäminen

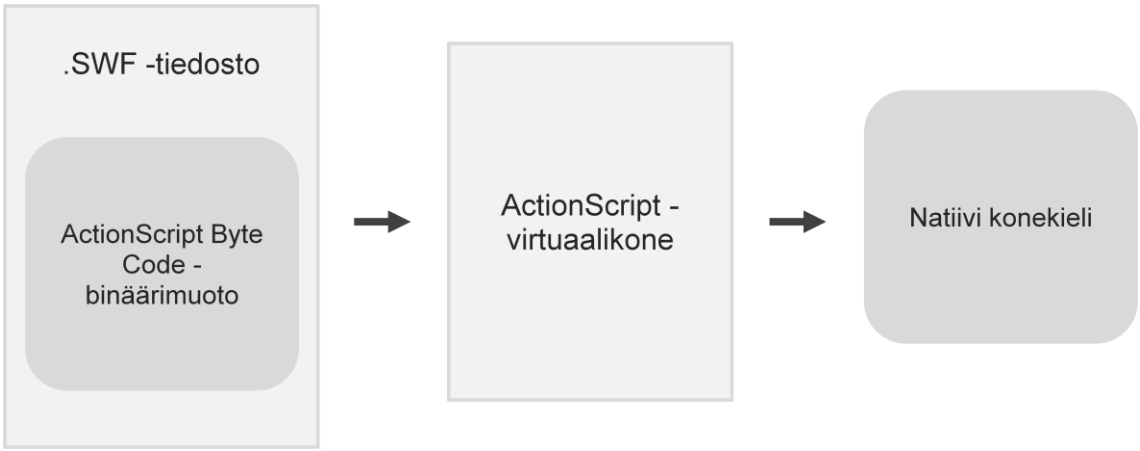
AIR-sovellusten toimintaperiaate Android-laitteissa on melko samankaltainen kuin työpöytäsovelluksissa. Laitteeseen asennetaan Adobe AIR -ajoympäristö, joka mahdollistaa Flash-tekniikalla toteutettujen sovellusten ajamisen laitteessa natiivisovellusten kaltaisina.

Julkaistaessa Flash-tekniikalla toteutettua sovellusta Android-laitteelle, projektin lähdekoodit ja muut siihen sidotut lähdetiedostot käännetään binäärimuotoon. Binäärimuotoinen data, jota kutsutaan nimellä ActionScript Byte Code, paketoidaan tämän jälkeen SWF-tiedostoon (kuvio 5). Lopuksi SWF-tiedosto ja muut ulkoiset resurssitiedostot paketoidaan APK-tiedostoksi, josta sovellus voidaan asentaa Android-laitteeseen (Wagner 2011, 5).



KUVIO 5. Flash -tekniikalla toteutetun sovelluksen kääntämisprosessi Android -laitteelle (Wagner 2011, 6, muokattu).

Ajettaessa AIR-sovellusta Android-laitteessa ajoympäristön ActionScript -virtuaalikone prosessoi paketoidun SWF-tiedoston. Virtuaalikone lataa ActionScript Byte Code -binäärimuodon käyttömuistiin ja purkaa sen. Tämän jälkeen Adobe AIR -ajoympäristö ajaa binääridatan tulkin läpi ja suorittaa sen natiivina konekielenä (kuvio 6).



KUVIO 6. Adobe AIR -sovelluksen ajonaikainen prosessi Android-laitteessa (Wagner 2011, 6, muokattu).

3.4 Tietoturva

3.4.1 Yleistä

Perinteisesti käyttöjärjestelmä asettaa työpöytäsovelluksille oikeudet sallituille toiminnoille kulloinkin sisään kirjautuneen käyttäjän oikeuksien mukaan (Wagner 2011, 9). Tämä malli toimii, koska käyttäjä joutuu erikseen asentamaan sovelluksen ja ilmaisee täten luottavansa siihen. Työpöytäsovelluksilla on siis muun muassa oikeus lukea ja kirjoittaa tietoa lokaaliin tiedostojärjestelmään.

Selaimessa ajettavat Internet-sovellukset taas eivät vaadi erillistä toimintaa käyttäjältä asentaakseen itsensä ja ovat työpöytäsovelluksia alttiimpia ulkoiselle tunkeutujalle (Wagner 2011, 9). Täten Internet-sovellusten oikeudet ovat paljon rajatummat työpöytäsovelluksiin verrattuna. Pääsy lokaaliin tiedostojärjestelmään on rajoitettu, lisäksi Internet-perustaiset toiminnot ovat mahdollisia vain selaimen asettamissa rajoissa ja yhden verkkotoimialueen sisällä.

3.4.2 Adobe AIR tietoturvamalli

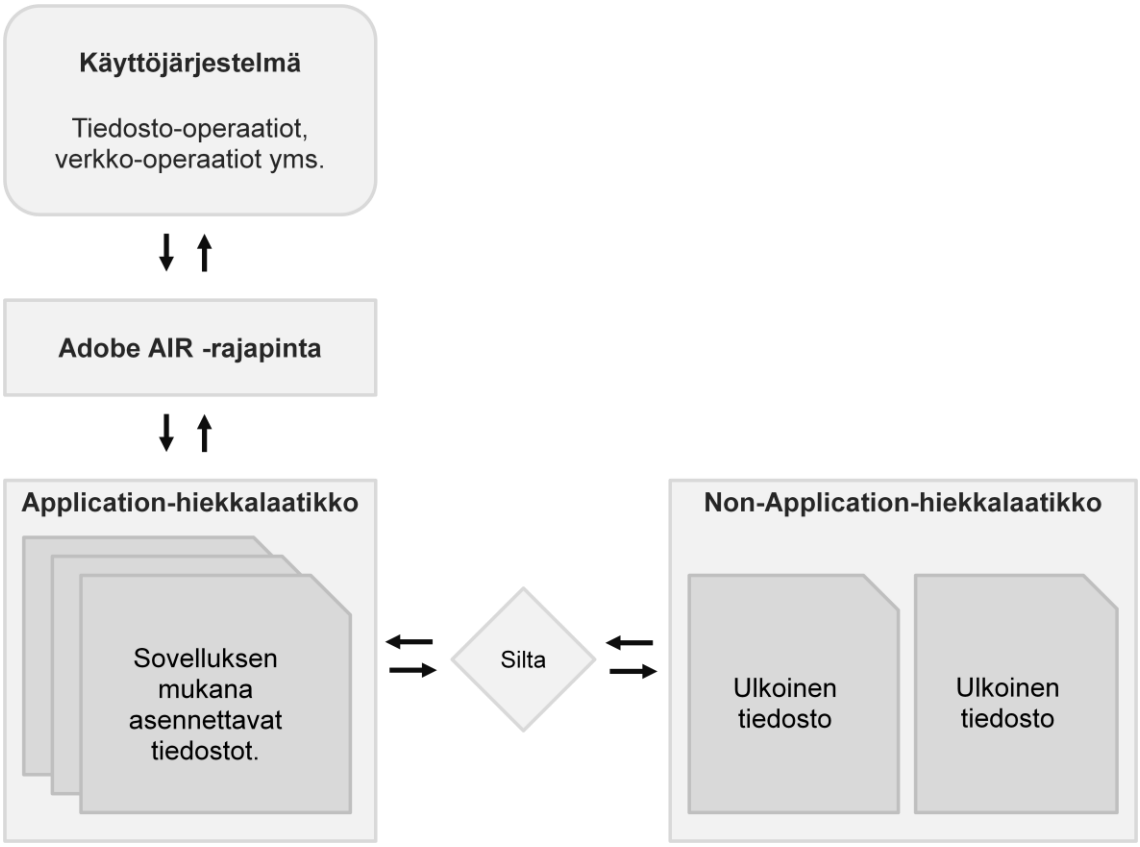
AIR-sovelluksiin pätevät samat rajoitukset kuin tavallisiin työpöytäsovelluksiin. Käyttöjärjestelmä siis rajoittaa sovelluksen oikeuksia muun muassa käyttäjän oikeuksien pojalta (Adobe AIR Security 2011, 1). Adobe AIR:n tietoturvamalli on kuitenkin eräänlainen välimuoto internet-sovellusten ja työpöytäsovellusten tietoturvamallien pohjalta.

Koska AIR-sovellukset hyödyntävät vahvasti internet-tekniikoita, jotka voivat mahdollistaa kolmannen osapuolen väliin tulemisen ja täten käyttäjälle vahingollisen toiminnan, on AIR-sovelluksilla tietoturvamalli, joka suojaa käyttäjiä tämänkaltaiselta toiminnalta (Wagner 2011, 10).

AIR-sovellus sallii oikeudet jokaiselle data- tai lähdetiedostolle erikseen, sen mukaan mikä on tiedoston alkuperä (Adobe AIR Security 2011, 6). Sovellus hyödyntää kahta erityyppistä ajonaikaista tietoturvalaatikkoa eriävine oikeuksineen ja asettaa jokaisen tiedoston toiseen tietoturvahiekkalaatikoista.

Tiedostot, jotka asennetaan AIR-sovelluksen mukana tämän omaan kotihakemistoon, sijoitetaan application-hiekkalaatikkoihin. Vain nämä tiedostot voivat hyödyntää AIR-tekniikan rajapintoja ja ajonaikaista ympäristöä.

Non-application-hiekkalaatikkoihin taas sijoitetaan kaikki tiedostot, jotka ovat joko kokonaan järjestelmän ulkopuolisia tai lokaaleja, mutta ulkona sovelluksen juurihakemistosta. Tämän hiekkalaatikon sisällä olevien tiedostojen oikeuksia rajoitetaan, eivätkä ne voi hyödyntää AIR-tekniikan rajapintoja (kuvio 7).



KUVIO 7. Adobe AIR -sovelluksen hiekkalaatikoiden avulla toteutettu tietoturvamalli (Wagner 2011, muokattu).

3.4.3 Digitaalinen allekirjoitus

Kaikki Adobe AIR -tekniikalla toteutetut sovellukset täytyy allekirjoittaa digitaalisesti. Digitaalisella allekirjoituksella varmistetaan sovelluksen julkaisija, sekä itse sovelluksen eheys. Kehittäjät voivat allekirjoittaa AIR-sovelluksensa joko luotettavan tahon myöntämällä, tai itse luomallaan sertifikaatilla (Adobe AIR Security 2011, 4-5). Käytettäessä luotettavan tahon digitaalista sertifiointia, voidaan julkaisija varmistaa sovellusta asennettaessa. Jos taas sovellus käyttää kehittäjän itse luomaa sertifikaattia, varoittaa AIR -ajoympäristö asennuksen yhteydessä käyttäjää siitä, ettei sovelluksen julkaisijaa voida varmistaa. Digitaalista allekirjoittamista ja sertifikaatin luomista käsitellään tarkemmin luvussa 4.

3.4.4 AIR-sovelluksen asentaminen ja päivittäminen

Adobe AIR -tekniikalla toteutetut työpöytäsovellukset levitetään ajoympäristön omina .air-muotoisina asennustiedostoina. Ajoympäristö vastaa sovelluksen asentamisesta koneelle, eikä kehittäjä voi vaikuttaa asennusprosessin vaiheisiin (Adobe AIR Security 2011, 1). Android-laitteille toteutetut AIR-sovellukset asennetaan tavallisten Android-sovellusten tavoin Android Market -sovelluskaupan kautta.

Aikaisemmin ajoympäristö täytyi asentaa erikseen, mutta kirjoitushetkellä uusimman version uutena ominaisuutena tuli mahdollisuus julkaista sovellus ja ajoympäristö myös yhtenäisenä asennuspakettina. Myös Adobe AIR -sovellusten ja itse ajoympäristön päivittäminen Android-laitteilla tapahtuu Android Market -sovelluskaupan kautta.

4 KÄYTTÖÖNOTTO JA KEHITYS

4.1 Kehitystyökalut

Kehitettiinpä AIR for Android -sovellusta sitten Adoben omilla kehitysympäristöillä, tai vaikkapa ilmaisella ja avoimen lähdekoodin FlashDevelop-sovelluksella, tarvitaan Android-sovelluksen asentamiseen myös Androidin omia kehitystyökaluja. Tässä luvussa esittelen AIR for Android -kehityksessä käytettäviä työkaluja ja niiden toimintaa. Tämän jälkeen käyn sovelluksen julkaisuprosessin tarkemmin läpi Adobe Flash Professional CS5.5 -kehitysympäristöllä.

4.1.1 Adobe AIR SDK

Adobe AIR SDK on oleellinen työkalu AIR-sovellusten kehittämiseen. Uusin versio AIR SDK:sta löytyy Adoben sivustolta, osoitteesta: http://www.adobe.com/special/products/air/sdk/. Adobe AIR SDK tarjoaa ohjelmointirajapintojen lisäksi kaksi hyödyllistä komentorivityökalua AIR -sovelluksen kehitykseen.

Adobe AIR Debug Launcher

AIR Debug Launcher (ADL) mahdollistaa AIR-sovelluksen suorittamisen ilman sovelluksen erillistä paketointia ja asentamista. ADL mahdollistaa myös sovelluksen yksinkertaisen testaamisen tulostamalla trace-lausekkeet ja ajonaikaiset virheet (Building Adobe AIR Applications 2011, 138). ADL:n avulla on täten mahdollista suorittaa ja testata myös kolmannen osapuolen kehitystyökaluilla ohjelmoituja AIR-sovelluksia.

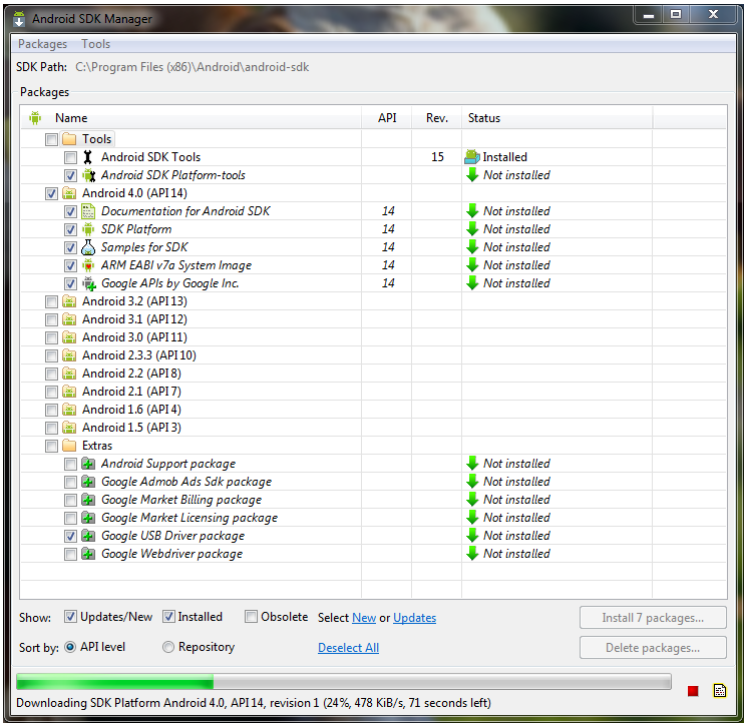
Adobe AIR Developer Tool

Air Developer Tool (ADT) on monikäyttöinen Java-sovellus, joka mahdollistaa muun muassa AIR-sovelluksen paketoimisen, sekä digitaalisen sertifikaatin luomisen ja sen allekirjoittamisen. Tämän lisäksi ADT mahdollistaa sekä AIR -ajoympäristön että AIR-sovelluksen etäasentamisen, -ajamisen ja -poistamisen mobiililaitteessa. Koska ADT on toteutettu Javalla, tarvitsee se toimiakseen myös vähintään Java 1.5 -asennuksen (Building Adobe AIR Applications 2011, 143).

4.1.2 Android SDK

Android SDK ei ole pakollinen valmiin ja käännetyn AIR for Android -sovelluksen toteuttamiseksi, mutta sitä tarvitaan sovelluksen asentamiseksi ja testaamiseksi Android-laitteessa tai työpöydällä ajettavassa emulaattorissa. Uusimman version Android SDK:sta voi ladata osoitteesta: http://developer.android.com/sdk/index.html. Android SDK vaatii toimiakseen myös Java SE JDK -paketin, joka on mahdollista ladata osoitteesta: http://www.oracle.com/technetwork/java/javase/downloads/index.html.

Android SDK tarjoaa oman Android SDK Manager -hallintasovelluksen eri työkalujen ja komponenttien asentamiseen ja päivittämiseen (kuvio 8). Android SDK Manager -sovellus sisältää myös AVD Manager -työkalun, jonka avulla on mahdollista luoda ja hallita työpöydällä ajettavia Android-emulaattoreita.



KUVIO 8. Android SDK Manager on sovellus eri työkalujen ja Android SDK-komponenttien hallintaan.

Yksi Android SDK:n hyödyllisimmistä työkaluista on Android Debug Bridge (ADB), joka mahdollistaa AIR for Android -sovellusten asentamisen ja ajamisen joko Android-laitteessa, tai AVD Manager -sovelluksella luodussa Android -emulaattorissa. Tämän lisäksi ADB kykenee siirtämään tiedostoja tietokoneen ja Emulaattori-instanssin välillä (Android Dev Guide, 2011).

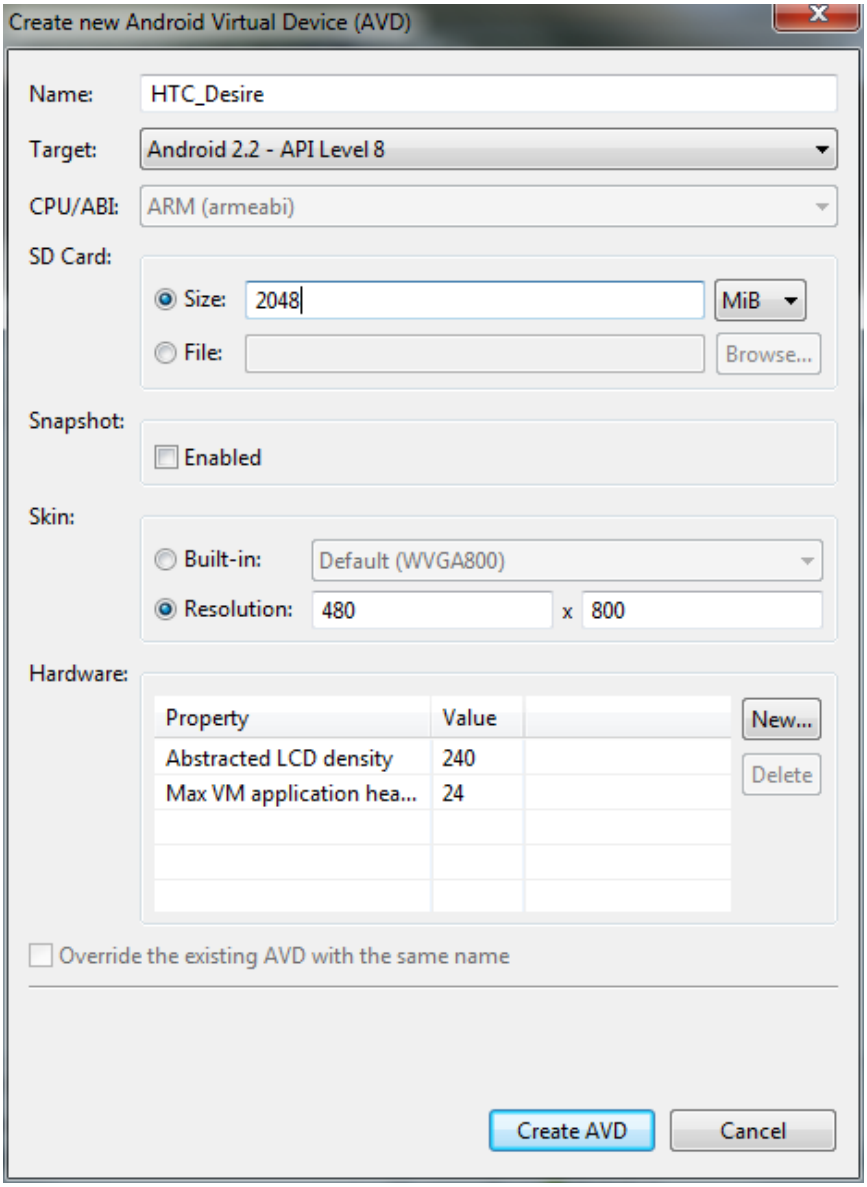
4.1.3 Flex SDK

Myös esimerkiksi Adobe Flex Builder -sovelluksen mukana tuleva Flex SDK mahdollistaa APK-tiedostojen paketoimisen ja täten AIR for Android -sovellusten kehittämisen (Wagner 2011, s. 22). Flex SDK:ta voidaan myös käyttää erikseen esimerkiksi jonkin kolmannen osapuolen ohjelmointiympäristön kanssa. Uusimman version avoimen lähdekoodin Flex SDK:sta voi ladata osoitteesta: http://opensource.adobe.com/wiki/display/flexsdk/Flex+SDK.

4.1.4 Android-emulaattori

Android-emulaattori mahdollistaa AIR for Android -sovelluksen testaamisen ilman Android-laitetta. Emulaattori on varsin hyödyllinen myös erilaisten laitekokoonpanojen - kuten vaikka eri näyttöresoluutioiden - testaamiseen. Emulaattori-instanssin, tarkemmin AVD:n (engl. Android Virtual Device) luominen tapahtuu jo aikaisemmin mainitulla AVD Manager -työkalulla. Uutta virtuaalilaitetta luotaessa on mahdollisuus määrittää muun muassa laitteen käyttöjärjestelmän versio, muistikortin koko, näytön resoluutio sekä lukuisia muita laiteasetuksia - esimerkiksi tukeeko laite GPS-paikannusta (kuvio 9).

Luodut AVD:t jäävät asetuksineen talteen, ja ne on täten helppo käynnistää uudelleen AVD Manager -työkalun avulla. Uutta virtuaalilaitetta luotaessa on myös mahdollista valita, säilytetäänkö laitteen tila tallentamalla tilannevedos AVD:sta. Tällöin virtuaalilaitetta suljettaessa sen tila tallennetaan levykuvaksi, joka voidaan palauttaa jälleen käynnistyksen yhteydessä.



KUVIO 9. Uutta AVD:tä luotaessa voidaan vaikuttaa virtuaalilaitteen lukuisiin kokoonpanoasetuksiin.

4.1.5 Kehitysympäristöt

AIR for Android -sovelluksia on mahdollista kehittää lukuisilla eri kehitysympäristöillä. Pelkästään Adobe tarjoaa jo kolme erilaista kehitysympäristöä sovelluksen kehittämiseen. Esittelen seuraavaksi merkittävimmät kehitysympäristöt, ja vertailen AIR for Android -kehitystä kyseisillä ympäristöillä.

Adobe Flash Professional CS5.5

Flash-sovellusten kehittäjille tutuin kehitysympäristö lienee Adoben Flash Professional, joka mahdollistaa monipuolisen AIR for Android -kehityksen ja sisältää monia työnkulkua helpottavia ominaisuuksia. Flash Professional mahdollistaa muun muassa AIR for Android -asetuksien muokkaamisen erillisestä asetus-ikkunasta. Android-sovelluksen pakkaaminen ja APK -tiedoston julkaiseminen, sekä sovelluksen asentaminen ja suorittaminen suoraan tietokoneeseen liitetyssä laitteessa on myös mahdollista Flash Professional -sovelluksesta käsin. Käsittelen AIR for Android -sovelluksen julkaisuprosessin Flash Professional -kehitysympäristöllä tarkemmin luvussa 5.

Adobe Flash Builder

AIR for Android -sovellusten kehittäminen on mahdollista myös Eclipse -kehitysympäristöön pohjautuvalla Adobe Flash Builder -sovelluksella. Sitä voidaan käyttää joko yhdessä Flash Professionalin kanssa, tai yksinään Flex SDK:n avulla. Käytettäessä samanaikaisesti Flash Professionalin kanssa, voidaan Flash Builder -ympäristöä käyttää saumattomasti lähdekoodin muokkaukseen ja Flash Professionalia esimerkiksi grafiikan ja aikajanan muokkaamiseen. Flash Builder -sovellusta voidaan käyttää myös ilman Flash Professional -sovellusta hyödyntämällä mukana tulevaa Flex SDK:ta sovellusten paketoimiseen ja kääntämiseen. Flash Builder versiosta 4.5 lähtien AIR for Android -sovelluksia on ollut mahdollista ohjelmoida myös käyttäen hyväksi Macromedian alunperin kehittämää MXML-merkkauskieltä käyttöliittymän luomisessa.

FlashDevelop

AIR for Android -sovelluksia on mahdollista kehittää myös ilman Adoben maksullisia kehitysympäristöjä. FlashDevelop on ilmainen avoimen lähdekoodin kehitysympäristö, joka tarjoaa tehokkaat työkalut AIR-sovellusten kehittämiseen Android-laitteille. FlashDevelop hyödyntää ilmaisia Flex ja AIR SDK:ta AIR for Android -sovellusten kehittämisessä, sekä käännettäessä sovellusta APK -asennuspaketiksi. Kirjoitushetkellä julkaistu versio 4.0 RC1 osaa tarvittaessa ladata ja asentaa sekä Flex SDK:n että AIR SDK:n automaattisesti ja hyödyntää niiden mukana tulevia komentorivityökaluja kehityksen tehostamisessa.

FlashDevelop hyödyntää erilaisia komentojonotiedostoja sovelluksen kääntämisessä, paketoimisessa ja laitteessa ajamisessa. Näiden komentojonotiedostojen avulla FlashDevelop mahdollistaa asetusten yksinkertaisen muokkaamisen ja sovellusten kääntämisen sekä suorittamisen suoraan kehitysympäristöstä käsin.

Omavalintainen tekstinmuokkausohjelma

Käytännössä AIR for Android -sovellus on mahdollista kehittää alusta loppuun saakka käyttämällä mitä tahansa tekstinmuokkausohjelmaa. Tällöin tosin joudutaan käyttämään käsin kaikkia SDK:n mukana tulevia komentorivityökaluja AIR for Android -sovellusten kääntämiseksi ja laitteessa testaamiseksi.

4.2 JULKAISUPROSESSI FLASH PROFESSIONAL -KEHITYSYMPÄRISTÖLLÄ

Tässä luvussa esittelen AIR for Android -sovelluksen julkaisuprosessin Adoben Flash Professional CS5.5 -kehitysympäristöllä. Käyn vaiheittain läpi tarvittavat komponentit ja niiden käyttämisen valmiin AIR for Android -sovelluksen julkaisemiseksi.

4.2.1 Digitaalisen sertifikaatin luominen

Kehittäjän itse allekirjoittamat sertifikaatit eivät juurikaan lisää sovelluksen luotettavuutta käyttäjien keskuudessa, koska mikään ulkopuolinen taho ei ole varmentanut allekirjoitusta. Tällainen Flash Professional -kehitysympäristöllä luotu sertifikaatti onkin tarkoitettu lähinnä sovelluksen testaamiseen itsenäisesti.

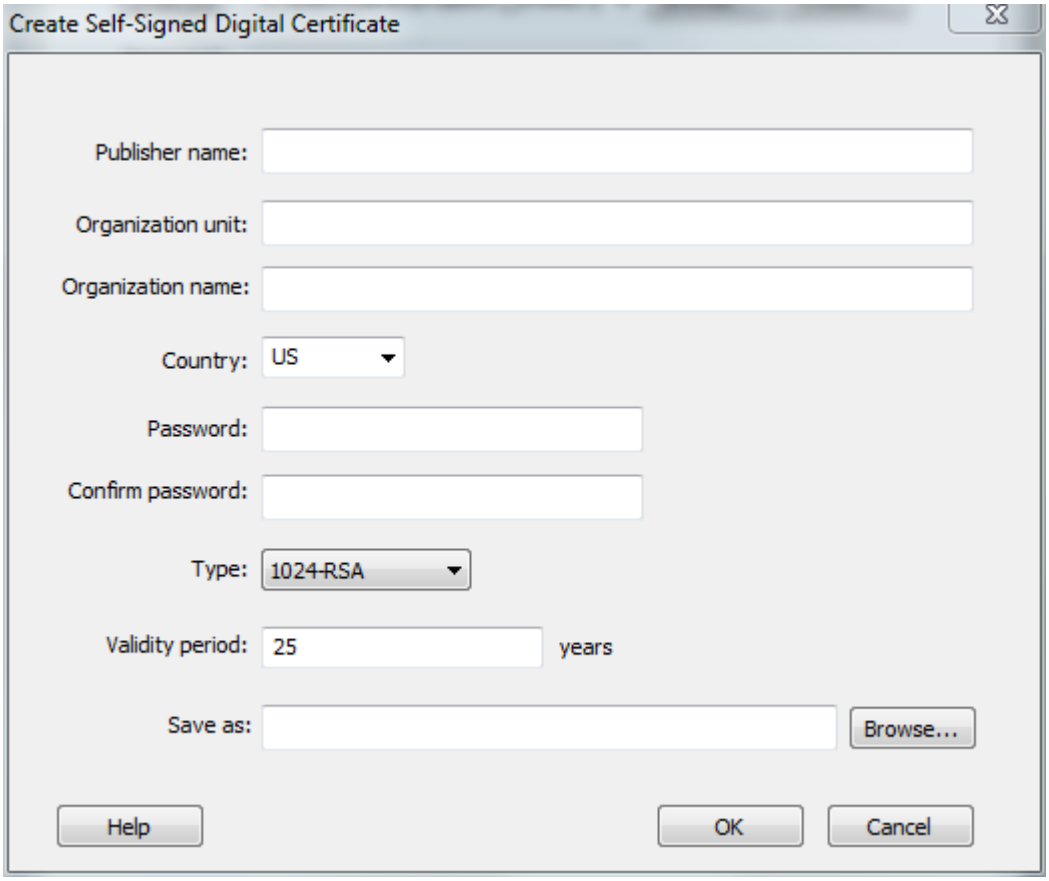
Digitaalisen sertifikaatin voi luoda Flash Professional -kehitysympäristön AIR for Android Settings -ikkunan Deployment-välilehdeltä. Uutta sertifikaattia luotaessa voidaan julkaisijan tietojen lisäksi valita sertifikaatin salasana, salausmetodi ja kuinka kauan luotu sertifikaatti on voimassa (kuvio 10).

Digitaalinen sertifikaatti on mahdollista luoda myös luvussa 4.1.1 mainittua AIR Developer Tool -komentorivityökalua käyttäen. Ohessa komennon syntaksi sertifikaatin luomiseksi (Wagner 2011, 16).

```
adt -certificate -cn yleisnimi salausmuoto tiedostonimi salasana
```

Esimerkiksi:

```
adt -certificate -cn cert1 1024-RSA mycert.p12 s414s4n4
```



KUVIO 10. Digitaalinen sertifikaatti voidaan luoda graafisen käyttöliittymän kautta suoraan Adoben Flash Professional -kehitysympäristöstä käsin.

Jokaiselle AIR for Android -sovellukselle ei tarvitse erikseen luoda omaa sertifikaattia, vaan kerran luotu sertifikaatti käy sekä työpöytä- että mobiilisovellusten digitaaliseen allekirjoittamiseen siihen asti kunnes sertifikaatin määritelty voimassaoloaika päättyy.

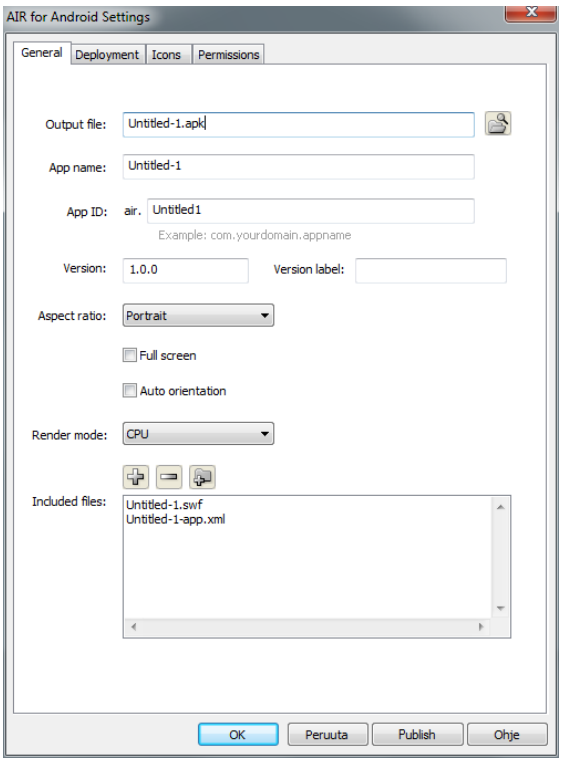
4.2.2 Android-asetusten määrittely

Jokainen Android-sovellus tarvitsee käännön yhteydessä erillisen XML -tiedoston, jossa määritellään sovelluksen komponentit ja muut keskeiset ominaisuudet, asetukset ja laiteominaisuudet, joita sovellus käyttää. Tällaista tiedostoa kutsutaan Manifest-tiedostoksi.

Adobe Flash Professional -kehitysympäristössä osaa Manifest-tiedoston asetuksista voidaan muokata helposti myös graafisen käyttöliittymän kautta, AIR for Android Settings -ikkunassa. Käyn seuraavaksi läpi eri asetukset välilehdittäin.

Yleistä-välilehti (General)

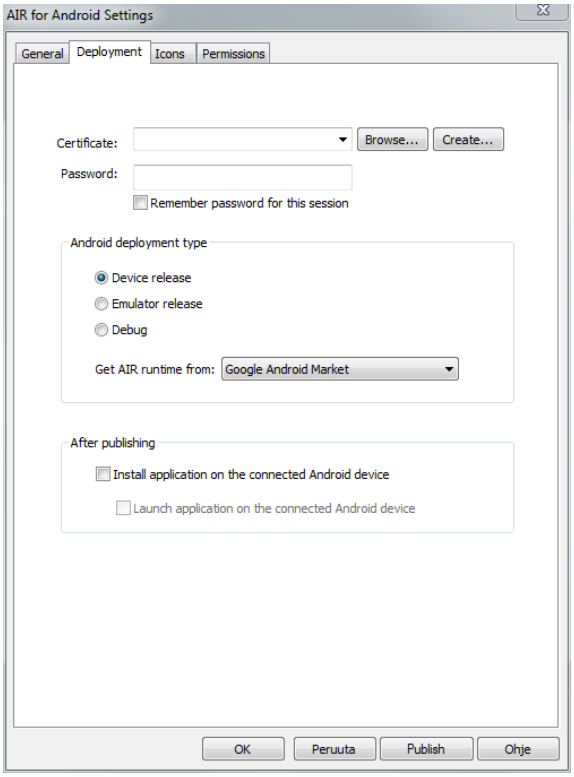
Ensimmäisellä välilehdellä voidaan määrittää sovelluksen nimi, tiedostonimi, tunniste, versionumero sekä version nimike. Tämän lisäksi yleistä-välilehdellä voidaan määrittää sovelluksen kuvasuhde; eli käynnistetäänkö sovellus pysty- vai vaakatilassa. Asetuksista voidaan myös valita, käynnistetäänkö sovellus koko näytön tilassa, ja orientoidaanko kuvasuhde automaattisesti sen mukaan, käytetäänkö sovellusta pysty- vai vaakatilassa. Lopuksi voidaan valita käytetäänkö sovelluksen graafiseen piirtämiseen laitteen prosessoria vai näytönohjainta, sekä mitä ulkoisia tiedostoja sovellukseen halutaan sisällyttää (kuvio 11).



KUVIO 11. Yleistä-välilehdellä voidaan vaikuttaa sovelluksen perusasetuksiin.

Käyttöönotto-välilehti (Deployment)

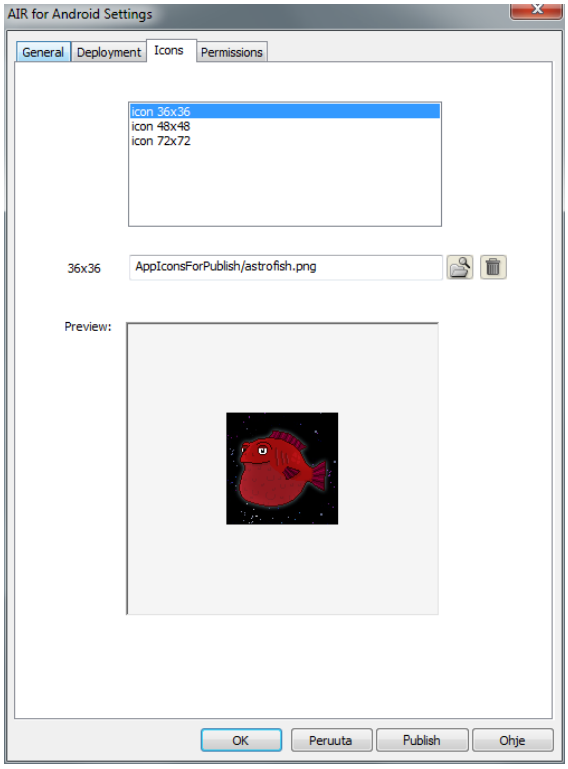
Käyttöönotto-välilehdellä voidaan vaikuttaa sovelluksen julkaisuasetuksiin, ja luoda esimerkiksi aikaisemmin mainittu itse allekirjoitettu digitaalinen sertifikaatti. Aluksi valitaan käytettävä digitaalinen sertifikaatti ja syötetään sertifikaatin salasana. Tämän jälkeen voidaan valita julkaisumuoto; eli julkaistaanko sovellus laitteeseen, emulaattoriin vai testaukseen. Lopuksi voidaan valita asennetaanko sovellus sitä julkaistaessa tietokoneeseen liitettyyn laitteeseen, ja suoritetaanko sovellus laitteessa (kuvio 12).



KUVIO 12. Käyttöönottoasetuksista valitaan sovelluksen julkaisuun liittyvät asetukset.

Kuvakkeet-välilehti (Icons)

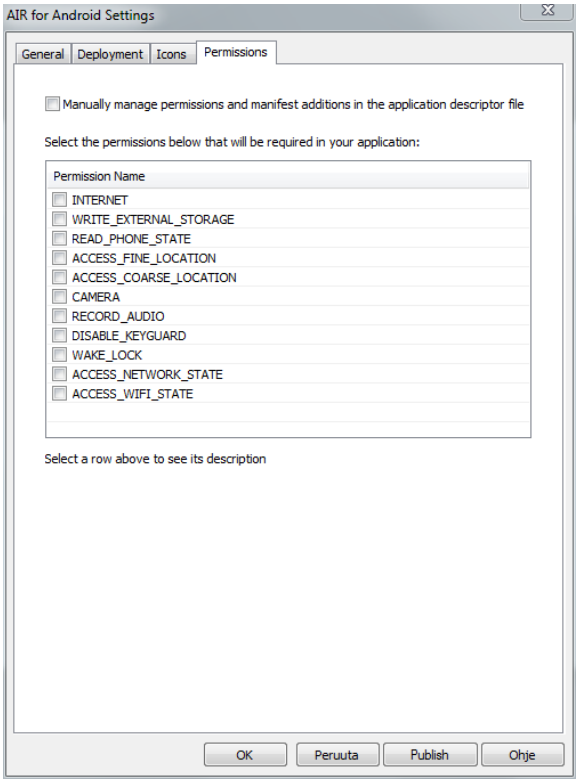
Kuvakkeet-välilehdeltä valitaan sovelluksen käyttämät kuvake-tiedostot. Kuvakkeista voidaan valita kolme eri versiota resoluution mukaan (kuvio 13). Jos kuvakkeita ei määritetä, Adobe Flash Professional käyttää omia oletusikoneitaan.



KUVIO 13. Kuvakkeet-välilehdeltä voidaan valita sovelluksen käyttämät kuvakkeet.

Oikeudet-välilehti (Permissions)

Oikeudet-välilehdeltä voidaan määrittää mitä oikeuksia ja laiteominaisuuksia sovellus pyytää käyttöönsä. Tällaisia ominaisuuksia voivat olla esimerkiksi Android-laitteen kamera, GPS-paikannin tai lupa käyttää laitteen Internet -yhteyttä (kuvio 14). Käyttäjä joutuu hyväksymään sovelluksen pyytämät oikeudet asentaessaan sovellusta laitteeseensa.



KUVIO 14. Oikeudet-välilehdeltä valitaan mitä käyttöoikeuksia sovellus pyytää Android-laitteelta.

5 JOHTOPÄÄTÖKSET JA POHDINTA

AIR for Android -sovelluskehitys on vielä suhteellisen uusi ilmiö, ja Adobe tuntuukin kehittävän tasaiseen tahtiin uusia ominaisuuksia ja siten myös uusia versioita AIR-ajoympäristöstään.

Vaikka monet tahot ovat julkisesti epäilleet Flash-teknologian poistuvan käytöstä muun muassa uuden HTML5-standardin alta, näyttää Flash- ja AIR -sovellusten tulevaisuus kuitenkin erittäin mielenkiintoiselta. Opinnäytetyön kirjoitushetkellä julkaistussa AIR-ajoympäristön 3.0-versiossa lisättiin tuki muun muassa natiivin konekielen laajennuksille. Tämä mahdollistaa myös sellaisten laite- ja alustaspesifien ominaisuuksien hyödyntämisen, jotka ovat aikaisemmin olleet vain natiivisovellusten käytössä.

Jos Flash-teknologia on entuudestaan tuttua, kynnys AIR for Android -sovelluskehitykselle on erittäin pieni. Ohjelmointiympäristön asentaminen ja käyttöönotto ei ole erityisen vaativaa, ja sovelluskehitystä on mahdollista tehdä myös avoimen lähdekoodin ilmaisilla ohjelmistoilla ja työkaluilla. Näkisinkin, että AIR for Android -sovelluskehitys voi olla erittäin kustannustehokas valinta tilanteessa, jossa pitäisi toteuttaa pieni tai keskisuuri monialustainen ohjelmisto- tai peliprojekti. AIR-teknologiaa käyttämällä samaa lähdekoodia voidaan hyödyntää niin sovelluksen selain-, työpöytä- kuin mobiiliversiossakin.

AIR for Android -sovelluskehitystä harkittaessa tulee kuitenkin ottaa huomioon AIR-sovellusten saavutettavuus. Sovellukset kun vaativat AIR-ajoympäristön asentamisen myös mobiililaitteilla, ja kirjoitushetkellä 24 Megatavun kokoinen asennuspaketti voi olla liikaa käyttäjille, joiden Android-laitteen sisäinen muisti on rajallinen.

LÄHTEET

Adobe Systems Inc. Adobe AIR 3 / Frequently Asked Questions. 2011. Luettu 15.10.2011. http://www.adobe.com/products/air/faq.html.

Adobe Systems Inc. Adobe AIR Security. 2011. Päivitetty 2.5.2011. Luettu 10.10.2011. http://help.adobe.com/en_US/air/security/air_security.pdf.

Adobe Systems Inc. Building Adobe AIR Applications. 2011. Päivitetty 14.10.2011. Luettu 15.10.2011. http://help.adobe.com/en_US/air/build/air_buildingapps.pdf.

Adobe Systems Inc. Flash Platform for TV. 2011. Luettu 14.10.2011. http://www.adobe.com/devnet/devices/flash_platform_tv.html.

Allaire, J. 2002. Macromedia Flash MX-A next-generation rich client. San Francisco: Macromedia Inc.

Chambers, M., Dura, D., Georgita, D. & Hoyt, K. 2008. Adobe AIR for Javascript Developers Pocket Guide. Sebastopol: O'Reilly Media Inc.

Distinctive Features of SQLite. 2011. Luettu 14.10.2011. http://www.sqlite.org/different.html.

Garret, J. 2005. Ajax: A New Approach to Web Applications. Luettu 29.9.2011. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications.

Google Inc. Android Dev Guide. Tools - Android Debug Bridge. 2001. Luettu 20.10.2011. http://developer.android.com/guide/developing/tools/adb.html.

Rich Internet Application Market Share. RIA Market Penetration and Global Usage. 2011. Luettu 10.10.2011. http://www.statowl.com/custom_ria_market_penetration.php

Wagner, R. 2011. Professional Flash Mobile Development. Creating Android and iPhone Applications. Indianapolis: Wiley Publishing Inc.